Software Model Checking for Verifying Distributed Algorithms

Sagar Chaki, James Edmondson
October 28, 2014

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Report Documentation Page

Form Approved
OMB No. 0704-0188

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE **28 OCT 2014**	2. REPORT TYPE **N/A**	3. DATES COVERED
4. TITLE AND SUBTITLE **Software Model Checking for Verifying Distributed Algorithms**		5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) **Chaki /Sagar**		5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S) 11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited.

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**SAR**	**16**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

DM-0001784

Motivation

Distributed algorithms have always been important

- File Systems, Resource Allocation, Internet, …

Increasingly becoming safety-critical

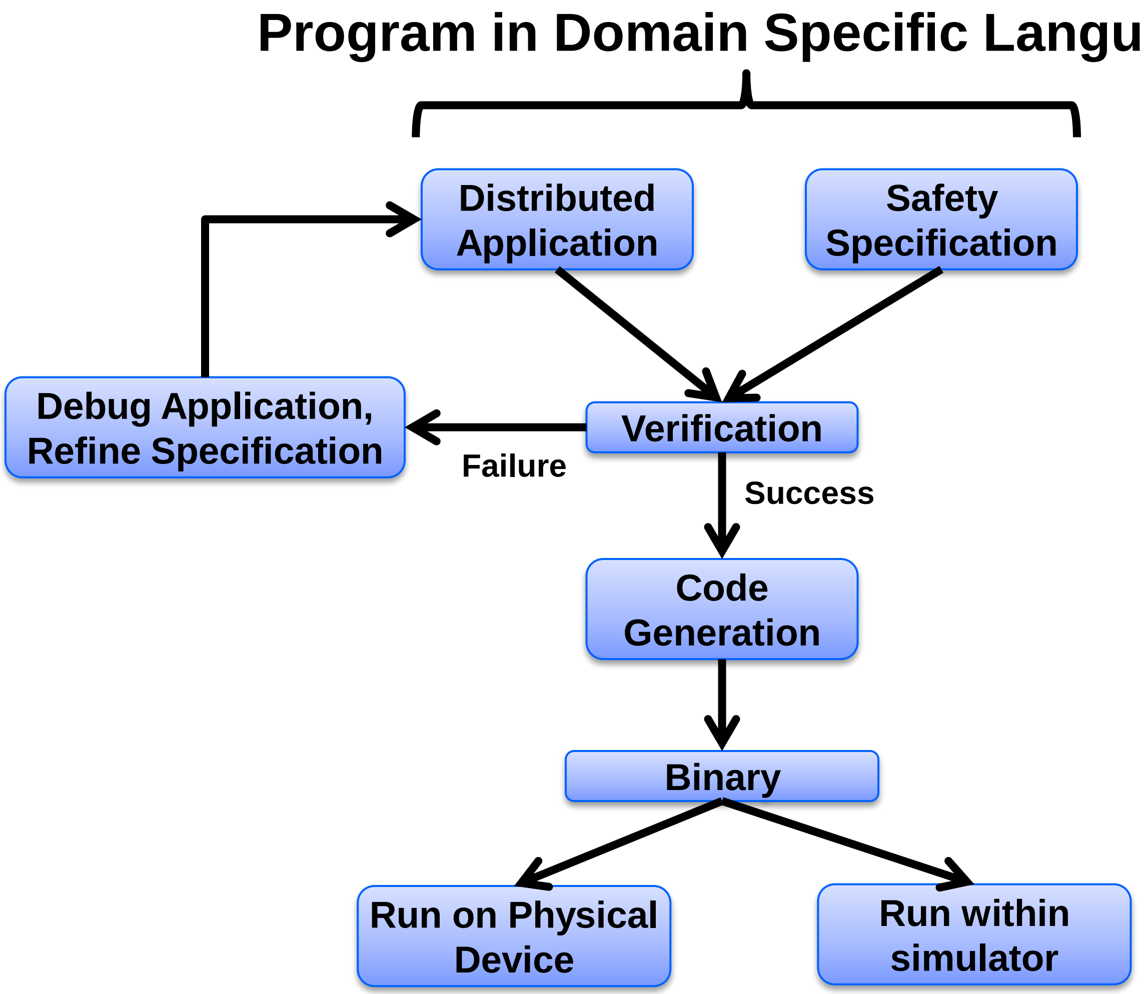
- Robotic, transportation, energy, medical

Prove correctness of distributed algorithm implementations

- Pseudo-code is verified manually (semantic gap)
- Implementations are heavily tested (low coverage)

Model-Driven Verifying Compilation of Synchronous Distributed Applications, Sagar Chaki, James Edmondson, Proc. of MODELS 2014, to appear

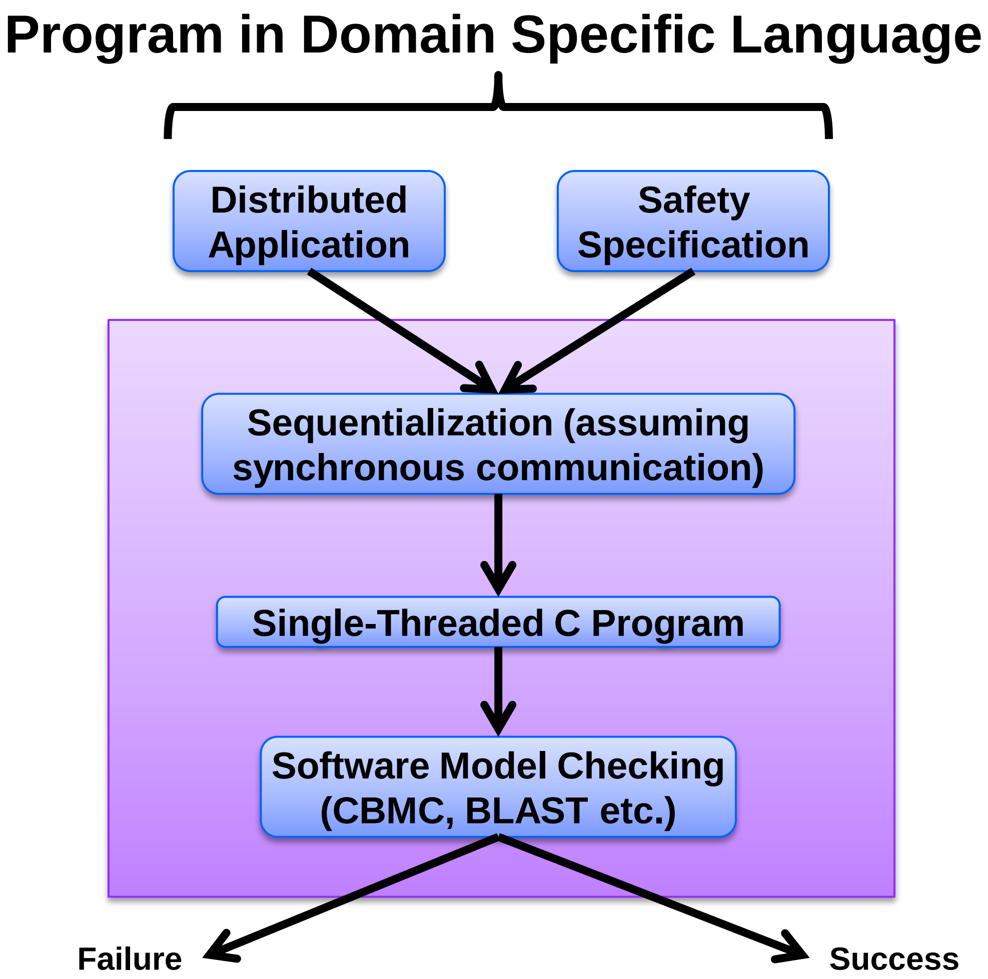
Approach : Verification + Code Generation



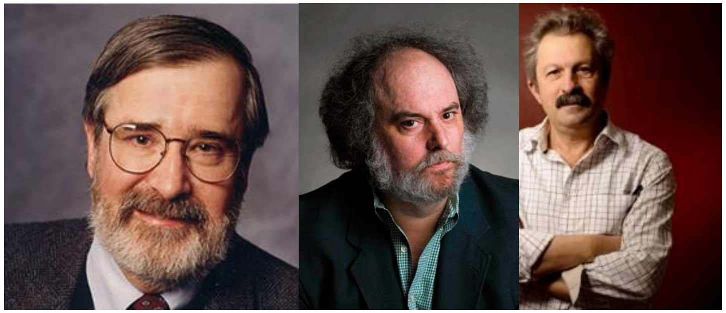
The Verifying Compiler: A Grand Challenge for computing research

Tony Hoare

Verification



Model Checking



Automatic verification technique for finite state concurrent systems.

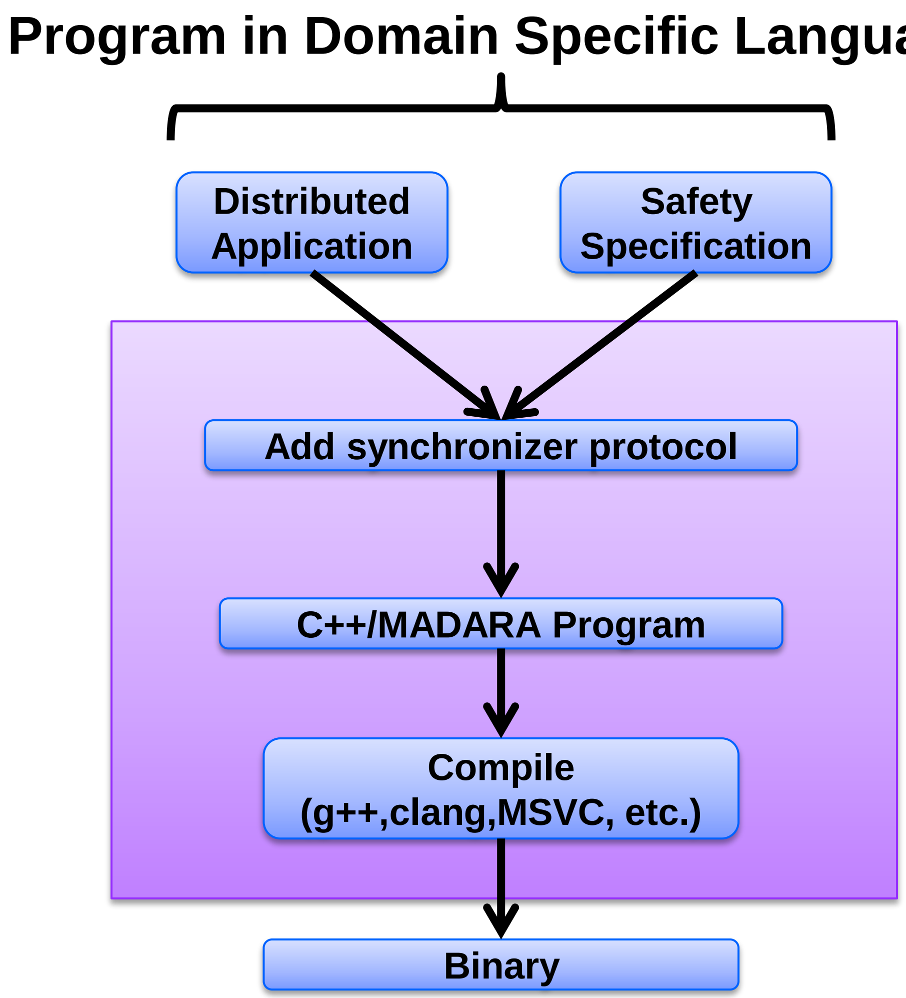
- Developed independently by Clarke and Emerson and by Queille and Sifakis in early 1980's.
- ACM Turing Award 2007

Specifications are written in propositional temporal logic. (Pnueli 77)

- Computation Tree Logic (CTL), Linear Temporal Logic (LTL), …

Verification procedure is an intelligent exhaustive search of the state space of the design

Code Generation



MADARA Middleware

A database of facts: $DB = Var \mapsto Value$

Node i has a local copy: DB_i

- update DB_i arbitrarily
- publish new variable mappings
  - Immediate or delayed
  - Multiple variable mappings transmitted atomically

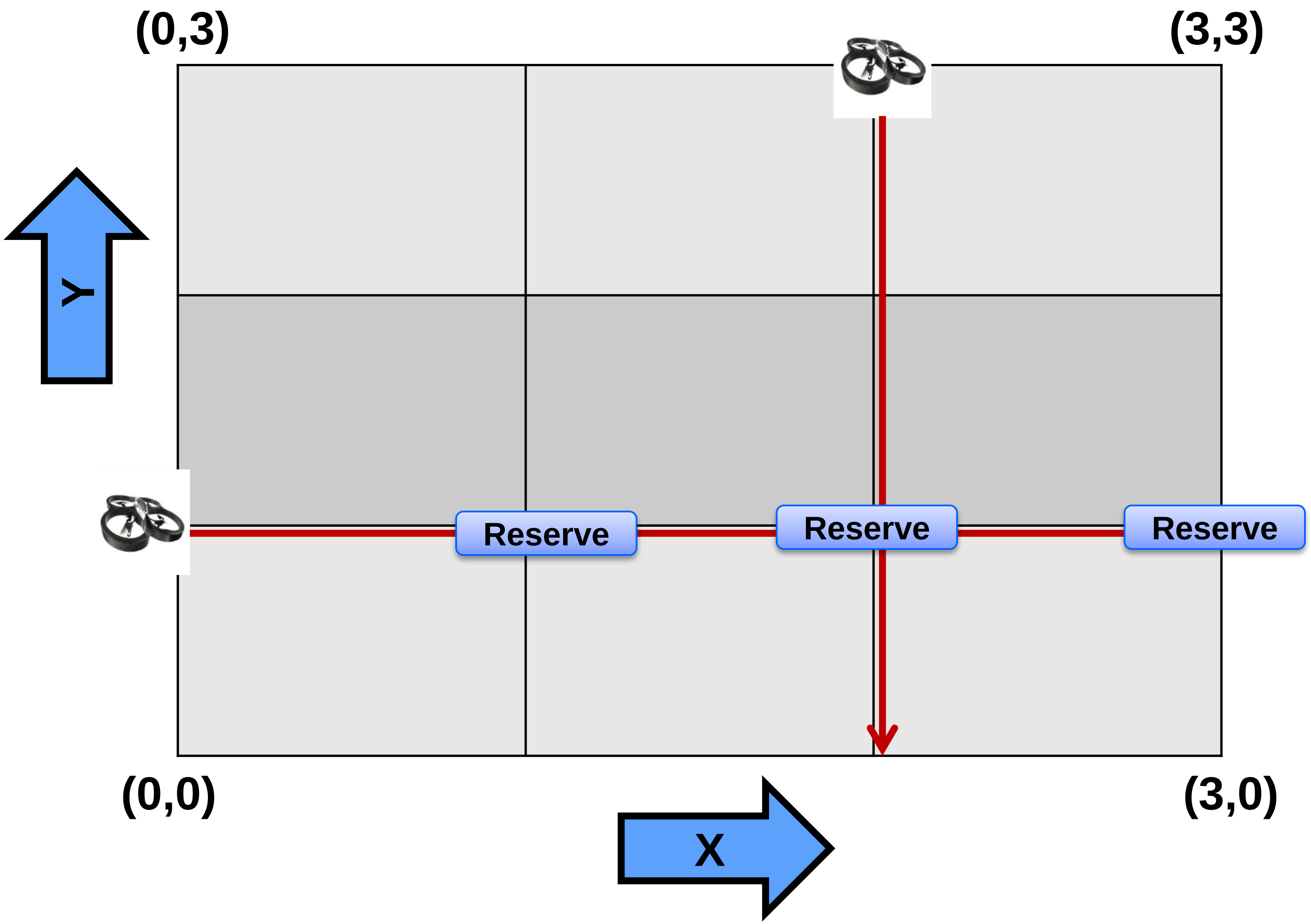
Implicit “receive” thread on each node

- Receives and processes variable updates from other nodes
- Updates ordered via Lamport clocks

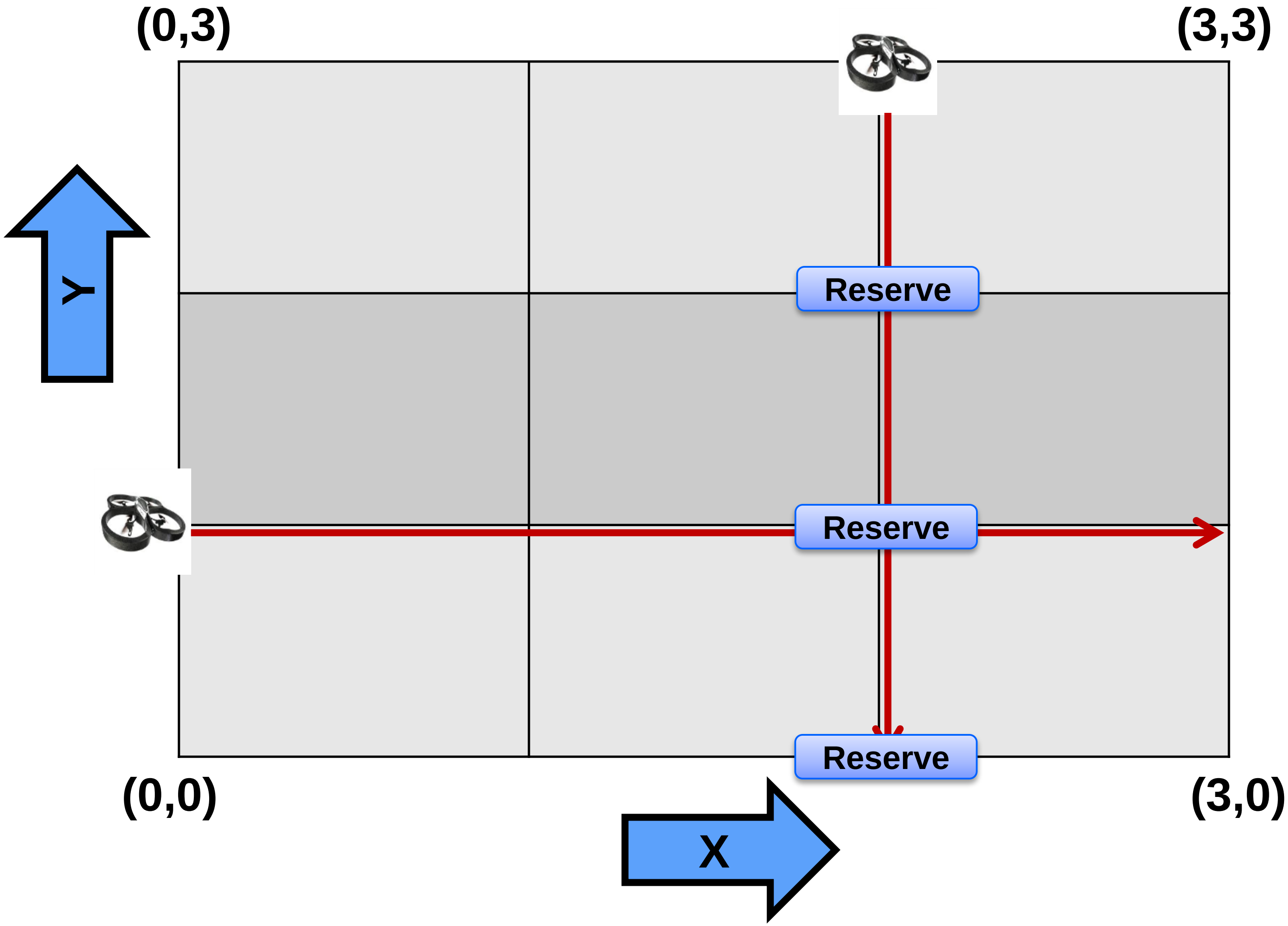
Portable to different OSes (Windows, Linux, Android etc.) and networking technology (TCP/IP, UDP, DDS etc.)

Case Study: Synchronous Collision Avoidance

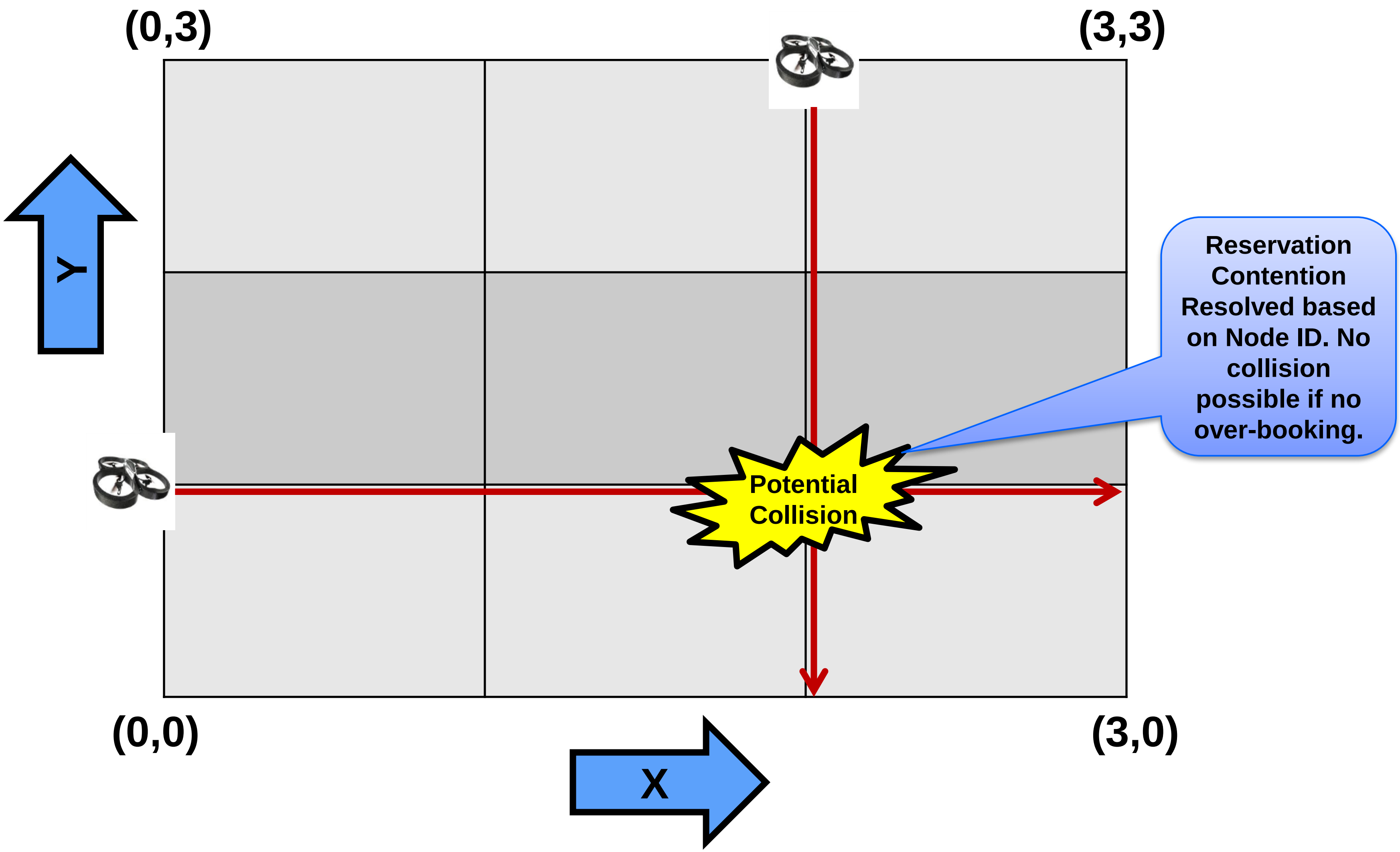
Example: Synchronous Collision Avoidance



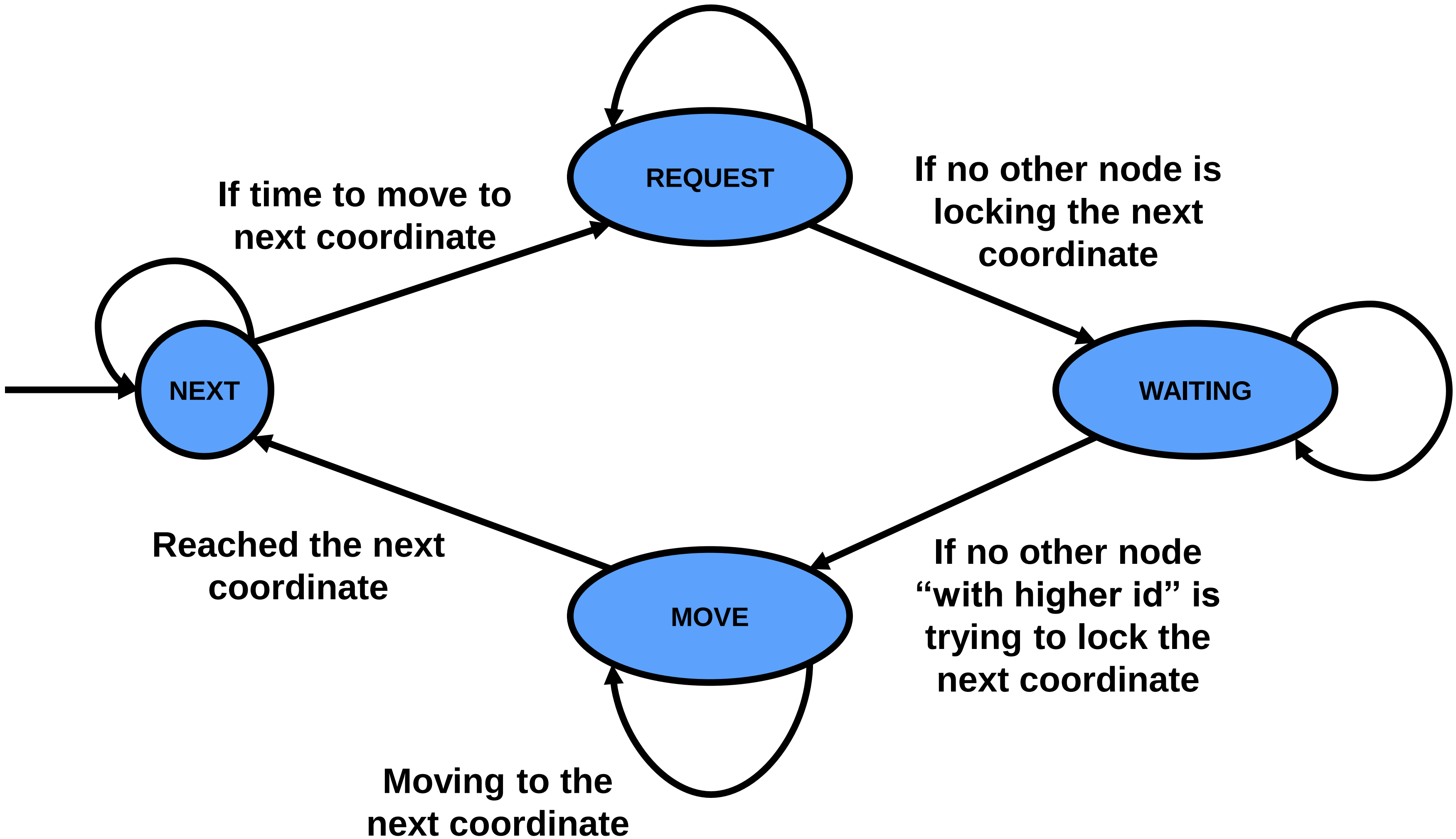
Example: Synchronous Collision Avoidance



Example: Synchronous Collision Avoidance



Collision Avoidance Protocol



Synchronous Collision Avoidance Code

```
MOC_SYNC;

CONST X = 4; CONST Y = 4;
CONST NEXT = 0;
CONST REQUEST = 1;
CONST WAITING = 2;
CONST MOVE = 3;

EXTERN int
MOVE_TO (unsigned char x,
         unsigned char y);

NODE uav (id) { … }

void INIT () { … }

void SAFETY { … }
```

```
NODE uav (id)
{
  GLOBAL bool lock [X][Y][#N];
  LOCAL int state,x,y,xp,yp,xf,yf;
  void NEXT_XY () { … }
  void ROUND () {
    if(state == NEXT) { …
      state = REQUEST;
    } else if(state == REQUEST) { …
      state = WAITING;
    } else if(state == WAITING) { …
      state = MOVE;
    } else if(state == MOVE) { …
      state = NEXT;
    } } }
```

```
INIT
{
  FORALL_NODE(id)
    state.id = NEXT;
    //assign x.id and y.id non-deterministically
    //assume they are within the correct range
    //assign lock[x.id][y.id][id] appropriately

  //nodes don't collide initially
  FORALL_DISTINCT_NODE_PAIR (id1,id2)
    ASSUME(x.id1 != x.id2 || y.id1 != y.id2);
}

SAFETY {
  FORALL_DISTINCT_NODE_PAIR (id1,id2)
    ASSERT(x.id1 != x.id2 || y.id1 != y.id2);
}
```

Synchronous Collision Avoidance Code

```
if(state == NEXT) {
  //compute next point on route
  if(x == xf && y == yf) return;
  NEXT_XY();
  state = REQUEST;
} else if(state == REQUEST) {
  //request the lock but only if it is free
  if(EXISTS_OTHER(idp,lock[xp][yp][idp] != 0)) return;
  lock[xp][yp][id] = 1;
  state = WAITING;
} else if(state == WAITING) {
  //grab the lock if we are the highest
  //id node to request or hold the lock
  if(EXISTS_HIGHER(idp, lock[xp][yp][idp] != 0)) return;
  state = MOVE;
}
```

```
else if(state == MOVE) {
  //now we have the lock on (xp,yp)
  if(MOVE_TO()) return;
  lock[x ][y][id] = 0;
  x = xp; y = yp;
  state = NEXT;
}
```

Tool Usage

Project webpage (http://mcda.googlecode.com)

- Tutorial (https://code.google.com/p/mcda/wiki/Tutorial)

Verification

- daslc --nodes 3 --seq --rounds 3 --seq-dbl --out tutorial-02.c tutorial-02.dasl
- cbmc tutorial-02.c (takes about 10s to verify)

Code generation & simulation

- daslc --nodes 3 --madara --vrep --out tutorial-02.cpp tutorial-02.dasl
- g++ …
- mcda-vrep.sh 3 outdir ./tutorial-02 …

Demonstration: Synchronous Collision Avoidance

Contact Information Slide Format

Sagar Chaki
Principal Researcher
SSD/CSC
Telephone: +1 412-268-1436
Email: chaki@sei.cmu.edu

U.S. Mail
Software Engineering Institute
Customer Relations
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
USA

Web
www.sei.cmu.edu
www.sei.cmu.edu/contact.cfm

Customer Relations
Email: info@sei.cmu.edu
Telephone: +1 412-268-5800
SEI Phone: +1 412-268-5800
SEI Fax: +1 412-268-6257